Kepler - Bug #3774

Hello World demo causes UI to become unresponsive and require restart

01/23/2009 11:59 AM - David Groulx

Status:	New	**Start date:**	01/23/2009
Priority:	Normal	**Due date:**	
Assignee:	Chad Berkley	**% Done:**	0%
Category:	documentation	**Estimated time:**	0.00 hour
Target version:	3.X.Y	**Spent time:**	0.00 hour
Bugzilla-Id:	3774		

Description

The basic network described in getting-started-guide.pd, pg 30 causes the UI to freeze up. This network consists of a String Constant connected to a Display actor with an SDF director. The only default parameter changed was the value of the String Constant. See specified URL for the workflow file I am using.

uname -a:
Darwin dhcp87.nceas.ucsb.edu 9.6.0 Darwin Kernel Version 9.6.0: Mon Nov 24 17:37:00 PST 2008; root:xnu-1228.9.59~1/RELEASE_I386 i386

java -version:
java version "1.5.0_16"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_16-b06-284)
Java HotSpot(TM) Client VM (build 1.5.0_16-133, mixed mode, sharing)

svn info:
URL: https://code.kepler-project.org/code/kepler/trunk/modules/build-area
Repository Root: https://code.kepler-project.org/code/kepler
Repository UUID: edc41a2b-3e5c-0410-9d3f-8540a70682f1
Revision: 16428
Node Kind: directory
Schedule: normal
Last Changed Author: welker
Last Changed Rev: 16426
Last Changed Date: 2009-01-22 14:33:48 -0800 (Thu, 22 Jan 2009)

History

#1 - 01/23/2009 12:28 PM - Shawn Bowers

This is happening because the "iterations" parameter of the SDF director is set to 0 in your workflow. In the actual demo workflow (under getting_started directory) this is set to 1 -- but probably this is not mentioned in the documentation/getting started guide?

BTW, you can also set the "firingCountLimit" of the String Constant actor to 1 as well (currently set to NONE). Either setting "iterations" or "firingCountLimit" will result in the desired behavior.

So, technically, this is not a bug in the system, but possibly a bug in the documentation. So, I'm changing it to a "documentation" bug ...

#2 - 01/23/2009 12:41 PM - David Groulx

The odd thing is that if you set up a different network with an SDF Director with interations set to zero, for example, a File Reader going into a text display, the UI does not freeze and you are able to stop it out of the loop. The problem is not a function of SDF Director's iterations set to 0, there is something else there.

Second, if the default configuration of a module leads the program to an unrecoverable state without any warning at all, I would very much call that a bug, not a documentation problem.

#3 - 01/23/2009 01:04 PM - Shawn Bowers

(In reply to comment #2)

> The odd thing is that if you set up a different network with an SDF Director with interations set to zero, for example, a File Reader going into a text display, the UI does not freeze and you are able to stop it out of the loop. The problem is not a function of SDF Director's iterations set to 0, there is something else there.

Yes -- this is because of the nature of the FileReader actor. Of course, the workflow you describe is very different, and here if you were to set the iterations parameter of SDF to 1 you probably would not get the behavior you desire.

> Second, if the default configuration of a module leads the program to an unrecoverable state without any warning at all, I would very much call that a bug, not a documentation problem.

You might enter a new bug for what you would like to see here. For instance, it sounds like you might want to change the default configuration of the String Constant actor to have the firingCountLimit be set to 1. Alternatively, this might be a larger issue in which the solution to the bug is that the default configuration of all actors should not result in an "unrecoverable state" within Kepler w.r.t. infinitely running workflows.

However, as I read your original bug report -- it sounds like you tried to recreate the workflow, and in doing so, didn't carry over the various parameter settings which led to the workflow running essentially in an infinite loop. However, the actual Hello World demo does not have this problem, which is why I think at least for this bug, it is a documentation issue.

#4 - 01/23/2009 01:18 PM - Matt Jones

I think it would be impossible to have each actor have a default state that never failed in any situation. Under some directors, any particular actor may work in its default state but not under a different director. This isn't to say that we couldn't improve the default state of various actors, but I think it would be impossible to guarantee correct operation in all permutations of directors and workflows.

#5 - 01/23/2009 01:34 PM - David Groulx

That's true that trying to guarantee that no possible combination of modules would fail is impossible.

IMO the quick and dirty fix would be to add an autosave on run option. If a workflow will crash, at least it won't take all your work down with it.

The better fix I suspect is to move the actual running of the workflow into a separate thread so that regardless of which modules decide to go haywire, the workflow is unable of capturing the UI and causing a problem. I've only just started looking at the code so I don't know how difficult/feasible this would be.

#6 - 01/23/2009 02:10 PM - Matt Jones

As I digest this more, I think I agree that there is a problem if it completely seized up the UI. Were you able to hit the 'Stop' button to stop the execution and regain control? If not, then that is probably a bug -- even when a workflow is in an infinite loop, the controls should still work to let it trigger execution to stop (although it does generally take a little bit for the director to let actors that are in mid-fire to complete their work).

#7 - 01/23/2009 02:23 PM - David Groulx

(In reply to comment #6)

> As I digest this more, I think I agree that there is a problem if it completely seized up the UI. Were you able to hit the 'Stop' button to stop the execution and regain control?

No, the stop button, nor anything in the UI for that matter, completely stops responding to user input.

#9 - 12/05/2011 06:41 PM - Christopher Brooks

Edward writes:

I think what is going on is that the display actor queues
up a large number of events for the swing thread to deal
with, and when you hit the stop button, that event just goes
to the end of the queue. The swing thread won't respond
to the stop until it has processed all the display events.

The only fix I can think of is to put a Thread.sleep()
in the display actor. Thread.yield() is not enough
(we are already doing that, I think). But Thread.sleep()
will introduce a noticeable performance hit...

What we really need is some mechanism to get to the head
of the swing event queue, or to cancel pending events...

#10 - 03/27/2013 02:24 PM - Redmine Admin

Original Bugzilla ID was 3774

Files

junk.xml	9.77 KB	12/06/2011	Christopher Brooks